Fusion-Based
Person Detection System

Student: Nirosha Fernando
Supervisor: Dr. Andrea Cavallaro
Report Number: 2008-AC-ee03026
Date: 21st April 2008
Course: MEng Computer Engineering

Department of Electronic Engineering
Disclaimer

“This report is submitted as part requirement for the degree of MEng Computer Engineering at the University of London. It is the product of my own labour except where indicated in the text. The report cannot be copied and distributed even if the source is acknowledged.”
Dedication

“This Report is dedicated to my parents.”
Acknowledgements

I am pleased to acknowledge the contribution of many people whose interaction, support and guidance leads to the accomplishment of this project. I would like to thank my supervisor, Dr Andrea Cavallaro for his full involvement in the project. I also thank the Computer Vision and Multimedia Research Group of QMUL and in particular Murtaza Taj for his support and help throughout the project. His experiences and work has a huge contribution towards this project.

I would like to acknowledge the work carried out by the following organisations/bodies responsible for the datasets used in this project, for their time and effort spent on compiling valuable data resources: DaimlerChrysler Pedestrian Benchmark, MIT CBCL, PETS, i-LIDS.

Finally, I would like to thank all my colleagues, friends and all the staff of Queen Mary University of London for providing me all required assistance, resources and facilities.
Contents

Abstract .......................................................................................................................... VII

1 Introduction .............................................................................................................. 1
  1.1 Motivation ......................................................................................................... 1
  1.2 Applications .................................................................................................... 1

2 Background ............................................................................................................. 2
  2.1 Features .......................................................................................................... 2
    2.1.1 Haar-like Rectangle Filters ...................................................................... 2
    2.1.2 Edgelets [7] ............................................................................................ 2
  2.2 AdaBoost ......................................................................................................... 3
    2.2.1 Discrete AdaBoost .................................................................................. 3
    2.2.2 Real AdaBoost ....................................................................................... 3
    2.2.3 Gentle AdaBoost .................................................................................... 4

3 Method ................................................................................................................... 5
  3.1 Cascaded Learning Algorithm ........................................................................... 5
  3.2 Cascaded Detector ........................................................................................... 7
  3.3 Dataset ............................................................................................................ 8
    3.3.1 DaimlerChrysler Pedestrian Classification Benchmark ......................... 8
    3.3.2 MIT CBCL Pedestrian Database ............................................................ 9
    3.3.3 Negative Data ........................................................................................ 9
    3.3.4 Final Dataset .......................................................................................... 10
  3.4 Motion Images ................................................................................................ 11
  3.5 Filter-set ......................................................................................................... 12
  3.6 Integral Image ................................................................................................ 13
  3.7 Training .......................................................................................................... 15
  3.8 AdaBoost ....................................................................................................... 17
Abstract

This project implements a pedestrian detection system using a dynamic classifier that ‘fuses’ together appearance and motion patterns. The cascaded classifier is built using AdaBoost learning algorithm, which picks up low-level Haar-like features by examining a large set of example images.

The classifier was trained using a frame-pair dataset adapted from the DaimlerChrysler Pedestrian Classification Benchmark dataset and MIT CBCL Pedestrian dataset. Validation was performed on frames obtained from video clips from, PETS 2001 and i-LIDS.

Performance of different AdaBoost versions were evaluated as well as numerous experiments with different combinations of filter-sets and datasets. The final detector scans a 320 by 240 pixel frame in approximately 37.6 seconds.
1 Introduction

1.1 Motivation
With the increased amount of surveillance systems being implemented around the world, the need for automated detection and tracking software has rapidly grown. Among these, person detection is an important task in many applications ranging from safety and security to leisure and sports. But this task is difficult due to variable appearance of the same object category as well as varying relation with the background. Various types of clothing, weather conditions as well as different styles of walking and other human gestures and poses make this a complex problem. Related work in the field suggests it is possible to detect these characteristics reliably when considered individually but requires lengthy analysis to achieve accuracy.

An alternative way of solving the problem is to fuse together information from multiple detectors such as appearance and motion. The various sources will work together to validate each other's results and eliminate false positives dramatically, without exhaustive analysis of individual features. The project aims to develop an algorithm that will use machine-learning techniques such as AdaBoost to build a classifier with minimal input from the user. The solution will be trained by a given dataset and then adapt itself to suit any scenario, regardless of scale or clothing.

1.2 Applications
The system can be used in many applications ranging from safety and security in surveillance to sports and recreation applications. A considerable amount of research has been carried out in developing pedestrian detectors by motor companies such as DaimlerChrysler, Ford and Mitsubishi, as part of a safety feature. The system could act as a base for accident prevention procedures such as automatic emergency breaks or automatic reduction in speed.

This can also be used in other transport related scenarios like trains and planes where the hazard is much higher. Detecting persons automatically in a high risk yet unlikely place such as a railway track or an airport runway could be very useful, since it eliminate the need for a human monitor who would have otherwise be looking at an empty screen most of the time.

Another important application could be to monitor a large number of surveillance cameras and alert a user only if a pedestrian is present. This is useful in scenarios where the number of cameras by far exceeds the number of screens available to display the feeds. A typical example is an underground metro station where all the corridors are monitored by hundreds of cameras but the actual surveillance is performed using a single individual using a few screens.

In sports, the focus could be placed on the players which can then be used for a number of applications. In a football match for example, the players could be streamed to a mobile phone with high resolution images while giving a very low resolution or discarding the background and the ball. The same concept can be used in web-based video conferencing where the person will be streamed with a higher quality than the background or any other moving object.
2 Background

2.1 Features

2.1.1 Haar-like Rectangle Filters

Figure 1. Rectangle Filter Types. Image Source: [1]

Rectangle filters produce differences of summed areas. These are extremely versatile as they are easy to implement, fast to execute and generate important information when used in specific positions and scales characteristic to the object class.

Types A and B can be useful to isolate regions of interest from the background. Type C filters are capable of detecting specific parts of a human such as a pair of legs or the head based on the aspect ratios and position. Type D filters represent diagonal filters and type E can detect curves.

These filters are popular for its computational speed, when implemented using ‘integral image’ [4] (Section 3.6).

2.1.2 Edgelets [7]

“An edgelet feature which is defined as a short segment of line or curve. Compared to haar-like feature it could be more flexible and precise to fit into the human contour.”

Figure 3. The structure of an edgelet feature

Image taken from [7]
2.2 AdaBoost

2.2.1 Discrete AdaBoost

Discrete AdaBoost (Freund & Schapire 1996)

1. Start with weights $w_i = 1/N$, $i = 1, \ldots, N$.

2. Repeat for $m = 1, 2, \ldots, M$:
   
   a) Fit the classifier $f_m(x) \in \{-1, 1\}$ using weights $w_i$ on the training data.
   
   b) Compute $e_m = E_w[1(y \neq f_m(x))]$, $c_m = \log((1 - e_m)/e_m)$.
   
   c) Set $w_i \leftarrow w_i \exp[c_m \cdot 1(y_i \neq f_m(x_i))], i = 1, 2, \ldots, N$, and renormalize so that $\sum_i w_i = 1$.

3. Output the classifier $\text{sign}\left[\sum_{m=1}^{M} c_m f_m(x)\right]$

Algorithm 1: $E_w$ represents expectation over the training data with weights $w = (w_1, w_2, \ldots, w_n)$. At each iteration AdaBoost increases the weights of the observations misclassified by $f_m(x)$ by a factor that depends on the weighted training error.

Figure 2. Discrete AdaBoost Algorithm [3]

2.2.2 Real AdaBoost

Real AdaBoost

1. Start with weights $w_i = 1/N$, $i = 1, 2, \ldots, N$.

2. Repeat for $m = 1, 2, \ldots, M$:
   
   a) Fit the class probability estimate $p_m(x) = \hat{P}_w(y = 1|x) \in [0, 1]$ using weights $w_i$ on the training data.
   
   b) Set $f_m(x) \leftarrow \frac{1}{2} \log \frac{p_m(x)}{1 - p_m(x)} \in R$.
   
   c) Set $w_i \leftarrow w_i \exp[-y_i f_m(x_i)], i = 1, 2, \ldots N$, and renormalize so that $\sum_i w_i = 1$.

3. Output the classifier $\text{sign}\left[\sum_{m=1}^{M} f_m(x)\right]$

Algorithm 2: The Real AdaBoost algorithm uses class probability estimates $p_m(x)$ to construct real-valued contributions $f_m(x)$.

Figure 3. Real AdaBoost Algorithm [3]
2.2.3 Gentle AdaBoost

**Gentle AdaBoost**

1. Start with weights $w_i = 1/N$, $i = 1, 2, \ldots, N$, $F(x) = 0$.

2. Repeat for $m = 1, 2, \ldots, M$:
   
   (a) Fit the regression function $f_m(x)$ by weighted least-squares of $y_i$ to $x_i$ with weights $w_i$.
   
   (b) Update $F(x) \leftarrow F(x) + f_m(x)$
   
   (c) Update $w_i \leftarrow w_i e^{-y_i f_m(x_i)}$ and renormalize.

3. Output the classifier $\text{sign}[F(x)] = \text{sign}[\sum_{m=1}^{M} f_m(x)]$

**Algorithm 4:** A modified version of the Real AdaBoost algorithm, using Newton stepping rather than exact optimization at each step

*Figure 4. Gentle AdaBoost Algorithm [3]*
3 Method

The framework is broadly based on work carried out by Viola et al. [1]. Accordingly, the aim is to develop a learning algorithm that will train a classifier by examining a large set of given examples. The learning algorithm will be based on AdaBoost [2][3]. Once the training stage is complete, persons on a given video clip can be detected using the learnt classifiers in a scale invariant manner.

The system has two main components: a cascaded learning algorithm that builds a dynamic classifier and a cascaded detector that discards non-pedestrians with minimal processing. The following sections describe the system components and stages in greater detail.

3.1 Cascaded Learning Algorithm

![System flow diagram of the cascaded learning algorithm](image)

At its starting point, the system consists of a collection of training data, associated with a weight distribution and a large feature pool. The training data consists of a large number of labelled example images and the feature pool contains various types of filters (described later) applied at all possible scales and positions. The weight distribution, which is initially uniform, indicates the importance/complexity of a given sample at each round. These three inputs are passed to the training algorithm, which returns a single filter. The selected filter with an appropriate threshold value will determine whether a given example is positive or negative. Each of these thresholded filters is called a feature and a thresholded sum of features makes up a classifier. The final detector uses multiple classifiers arranged in a cascade. Building this cascade of classifiers, including finding the optimal filters, their votes and all of the thresholds mentioned here is part of the learning algorithm’s task.
Learning begins by training the entire filter-set over the training dataset, together with the weight distribution. The filter responses are analysed to find the optimal feature (filter-threshold combination) that yields the lowest weighted error. This is the task of the training algorithm.

Next, the selected feature $F_i$ receives two separate ratings, $\alpha$ and $\beta$, based on its error for positive and negative classifications respectively. These values are the real-valued, confidence-based votes that a feature will output for a given sample.

$$F_i(I_t, I_{t+1}) = \begin{cases} \alpha & \text{if } f_i(I_t, \Delta, U, L, R, D) > t_i \\ \beta & \text{otherwise} \end{cases}$$

The weight of the samples misclassified by the selected feature is increased exponentially using the $\alpha$ and $\beta$ values. The new weights are normalised such that it is a distribution. This forces the training algorithm to focus on the misclassified samples that are now maximally difficult.

After each round of training, the validation step evaluates the performance of the current classifier by applying it to a set of validation frame-pairs. The threshold for the classifier is set such that very high detection rates are achieved while eliminating the false positives by half. More features are trained and added to the current classifier until both these requirements are satisfied. The resultant classifier $C$ is a linear combination of features, that outputs a binary decision based on a threshold $\theta$.

$$C(I_t, I_{t+1}) = \begin{cases} 1 & \text{if } \sum_{i=1}^{N} F_i(I_t, \Delta, U, L, R, D) > \theta \\ 0 & \text{otherwise} \end{cases}$$

Learning ends when the final targets for detection and false positive rates are achieved by adding a sufficient number of classifiers to the cascade $K$, with increasing amounts of features.

$$K = \prod_{n=1}^{N} C_n$$
3.2 Cascaded Detector

The detector takes as input, two frames from a given video and generates the motion images of the entire frame. The fixed-size detection window is then scanned over the integral image of the motion image frames, applying the cascade of classifiers at each location. Further stages of the cascade are not evaluated if a preceding stage classifies the candidate window as negative.

If all stages agree that the candidate window contains a person, the location and size of the detection is recorded and other locations are processed. Once the entire frame has been evaluated at all scales, K-means clustering eliminates most of the multiple detections around a single person. Finally the clustered detection bounding boxes are marked on the original frames and returned.
3.3 Dataset

Since the system uses empirical analysis to effectively ‘learn’ a classifier, there is a significant bearing on the design of the dataset. The dataset well dictate what to look for in a pedestrian and will set the limitations on the classifiers capabilities. If the dataset contains too many samples of the same pose or camera view, the classifier will be accurate for that case but restricted in others. If the dataset covers more scenarios, then the classifier may be too loose to detect any person, and may give rise to many false positives.

The negative examples are also of great significance, as it sets the contrast from the positive examples. Initial experiments with only static background used as negative samples proved to be less effective. The learnt classifier picked up almost all moving objects and missed stationary pedestrians. This included moving cars, carts, trees, etc. which implied the classifier was more of a motion detector rather than a person detector.

In addition, size and aspect ratios of the images are also important. The base scale will determine the smallest size of a person that can be detected and the aspect ratio will determine the shape of the detection window. This, in itself can act as a simple filter, as it implies the basic shape of an upright human.

Striking the right balance in all of these aspects will be a crucial step in building a good classifier. The dataset therefore needs to be engineered with planning and design, so that the images are carefully selected to produce the right results.

3.3.1 DaimlerChrysler Pedestrian Classification Benchmark

![Sample images from DC Pedestrian Classification Benchmark.](http://www.science.uva.nl/research/isla/downloads/pedestrians/assets/images/dc_ped_class_benchmark.gif)

The dataset was mainly sourced from DaimlerChrysler (DC) Pedestrian Classification Benchmark, which is available for use for non-commercial academic research purposes. The DC dataset contains cropped and centred images of pedestrians and non-pedestrians, resized to a common scale of 18 by 36 pixels. The images contained scenes from various places and times, with each original example followed by mirrored/shifted versions of the same person. Please refer to the dataset web page for a detailed description of the dataset.


However, since the DC dataset was designed to be used with a system that uses only appearance patterns, the dataset was not organised into consecutive frame-pairs. Adjacent frames from the same sequence were available for the positive samples but were scattered across the randomised dataset. The negative data mainly consisted of static backgrounds with no motion information. Furthermore, the size of the images was more than double compared to 15 by 20 pixels recommended by Viola, et al. This would generate more than 200,000 filters that can fit within the image.

Therefore, the following alterations were made in order to make the dataset suitable for this system:

- The positive images were manually re-organised into consecutive frame-pairs. Mirrored and shifted versions of the same example were removed.

- The images were re-sized to 12 by 24 pixels. This can be justified by considering the area of the images is about the same as 15 by 20 while retaining the original aspect ratios of the dataset.

- The negative data samples were duplicated to simulate static background patches. Minute brightness and contrast variations were added to the resultant frame-pairs.

### 3.3.2 MIT CBCL Pedestrian Database

An additional set of images were sourced from the Center for Biological & Computational Learning (CBCL) at MIT, which published a pedestrian dataset containing images of persons from a front and rear view. This dataset does not contain any motion information and therefore was used to simulate stationary persons in a typical scene. The original database contains colour images of size 64 by 128 in PPM format. Further information is available at MIT CBCL web site.¹

### 3.3.3 Negative Data

The negative data available from the datasets above, did not meet satisfactory standards to meet the requirements of this system, due to the lack of negative motion examples. Initial experiments did not produce intended results as described above. Therefore, additional negative data samples were added to the database, sourced from new video clips of streets scenes. The samples were extracted by scanning the actual

---

¹ MIT CBCL web site: [http://cbcl.mit.edu/projects/cbcl/software-datasets/PeopleData1Readme.html](http://cbcl.mit.edu/projects/cbcl/software-datasets/PeopleData1Readme.html)
detector, built using the classifier mentioned above that detected patches with motion, over full frames that did not contain persons. This produced valuable samples that are more relevant to the system and also added samples of various sizes that the final detector would actually pick up (resized to common size). The negative frames were mostly from street scenes with moving vehicles. There were also a few frames from video clips filmed during heavy snowfall. In addition, moving objects such as carts, baby prams, balls and birds were also manually extracted from video clips.

![Figure 9. Some of the frames used to extract negative examples. The image on the right is from a video clip filmed during a snowing.](image)

### 3.3.4 Final Dataset

The final dataset consists of 5280 example frame-pairs (10560 images), 2640 each of positive and negative samples. The images were 12 by 24 greyscale PGM type images, which takes only 324 bytes per sample. The images were sourced from the above sources in the manner detailed below.

Positive samples: 2640

- 1720 samples from DC dataset: Moving people at various poses with the detection window following the target between two frames.

- 900 samples from CBCL dataset: Original static images were duplicated, enlarged by two pixels and cropped to original size and then resized to common size of 12 by 24. This simulates person walking towards or away from the camera, making use of the front and rear view poses of this dataset.

- 20 New data samples: Frame-pairs containing persons from new video clips filmed at different times. The detection window is fixed between the two frames.

![Figure 10. Positive training data from the final dataset. Image-pairs on the left are from the DC dataset, while the image pairs derived from the MIT dataset are shown in the middle. The last image-pair is from a new video clip.](image)
Negative samples: 2640

- 1084 Negative motion samples: Extracted by scanning a basic detector over filmed video clips, as described above.
- 1556 samples from DC dataset: Original static images were duplicated as described above.

![Negative training data from the final dataset. The first three pairs of images were extracted by scanning negative frames, which contain moving parts of a cleaning trolley, a bus and a car. The next three were extracted from snowy scenes. The last two pairs are different types of windows from the DC negative dataset.](image)

### 3.4 Motion Images

The presence of motion in a video can be determined by calculating the absolute difference between two frames. In order to extract the key motion information from the frame-pairs, motion images are generated by taking the absolute differences of shifted images. That is, one of the images is shifted by a given amount (relative to the scale of the detection window) before calculating the difference.

There are five such motion images: $\Delta$ is the absolute difference between the original input images. U, D, L, R stand for the absolute difference between a shifted version of an input image and the shifted version of the second. Figure (9) shows some of the motion images produced by the system. Notice that the image with the lowest energy represents the direction in which an object is most likely moving [1].

![Examples motion images generated by the system](image)
3.5 Filter-set

The filter-set consists of the five types of rectangle filters described before, applied to the two original images and the five motion images shown above. When applied to the original input images, the filters are looking at static appearance patterns.

Detection of motion patterns is achieved using the same rectangle filters applied to motion images described above. In addition, single-box rectangle filters will then operate on shifted versions of the two frames to find the direction and magnitude of motion.

The four main categories of filters (similar to [1]) used are listed below:

**Appearance filters**

\[ f_a = \phi(I_t) \]

\( \phi \) is one of the five types of rectangle filters applied to one of the input images, \( I_t \).

**Motion shear filters**

\[ f_{ms} = \phi(S) \]

\( \phi \) is one of the five types of rectangle filters applied to one of the motion images, \( S \{U,D,L,R\} \) mentioned before.

**Motion direction filters**

\[ f_{md} = r_{md}(\Delta) - r_{md}(S) \]

\( r_{md} \) is a single-box rectangle filter applied to \( \Delta \) and one of the motion images, \( S \{U,D,L,R\} \).

**Motion magnitude filters**

\[ f_{mm} = r_{mm}(S) \]

\( r_{mm} \) is a single-box rectangle filter applied to one of the motion images, \( S \{U,D,L,R\} \).

The five rectangle filters applied to two static images and the five motion images as motion shear filters produce 35 types of filters. Motion direction filters and motion magnitude filters applied between the \( \Delta \) image and the other four motion images produce another eight types of filters. Therefore the number of filter-image combinations applicable at a given location reaches 43 (Although, not every filter maybe applicable at every location owing to incompatible aspect ratios).

When this is extended to all possible locations and scales, the total number of filters exceeds 100,000, which is not feasible to implement. Therefore the filter-set is sub-sampled by placing the origins of filters two pixels apart. The total size of the filter-set then is 33246 of which 7236 is static appearance filters and the rest is motion filters.
3.6 Integral Image

The speed of the rectangle filters described above, comes from calculating the ‘integral image’ prior to applying filters. This allows the filter values to be extracted from a fixed number of points, regardless of scale and position. Integral image is simply a matrix of the same size as the image, with each element holding the sum of all the pixels to the left and above itself. This can be calculated very quickly in one pass for the entire image.

![Integral Image](image.png)

The significance of the integral image becomes obvious by examining the way in which it is used to find the rectangle filter responses. The basic concept can be seen in a single box rectangle sum, as detailed below:

\[
X = BR - BL - TR + TL
\]
This can be extended to all other rectangle filter types:

**Type A Filters**

\[ A = \text{abs} \left( (D_1 + D_3 - D_2 - D_4) - (L_1 + L_3 - L_2 - L_4) \right) \]

Darker area

\[ \text{Lighter area} \]

**Type B Filters**

\[ B = \text{abs} \left( (D_1 + D_3 - D_2 - D_4) - (L_1 + L_3 - L_2 - L_4) \right) \]

Darker area

\[ \text{Lighter area} \]

**Type C Filters**

\[ C = \text{abs} \left( 2 \ast (D_1 + D_3 - D_2 - D_4) \right) \]

Darker area

\[ \text{Lighter area} \]

\[ - (L_1 + L_3 - L_2 - L_4) - (L_5 + L_7 - L_6 - L_8) \]

Note: The darker area is multiplied by two to compensate for the larger lighter area.

**Type D Filters**

\[ D = \text{abs} \left( (D_1 + D_3 - D_2 - D_4) + (D_5 + D_3 - D_2 - D_4) \right) \]

Darker area

\[ \text{Lighter area} \]

\[ - (L_1 + L_3 - L_2 - L_4) - (L_5 + L_7 - L_6 - L_8) \]

**Type E Filters**

\[ D = \text{abs} \left( (D_1 + D_3 + D_5 + D_7 - D_2 - D_4 - D_6 - D_8) \right) \]

Darker area

\[ \text{Lighter area} \]

\[ - (L_1 + L_3 + L_5 + L_7 - L_2 - L_4 - L_6 - L_8) \]
3.7 Training

The weak classifiers for the AdaBoost learning algorithm in relation to this scenario are features, made up of a filter and a threshold. Each feature makes the classification by applying the filter on a candidate image and checking the response against a suitable threshold for that filter. The task of the learning algorithm is to find the single feature that yields the least weighted error, based on the weight distribution of the training data.

**Optimal Threshold**

The responses for each filter are dependent on the type of filter, the area covered by the filter and the image that the filter applies to. As a result, the range of values for each filter is considerably different. Therefore, it makes sense to select the threshold values from the actual filter responses of the training samples. Given that the size of the filter-set is more than 30000 and considering that each of these filters may return up to 5280 different responses from each of the samples, the total number of filter-threshold combinations could be more than 175 million.

![Filter responses of four filters applied to a dataset of 2240 image-pairs. The x-axis represents the data sample index, where the first 1120 responses are from negative samples while the last half is from positive samples. Some filters produce responses that are easily distinguishable between positive and negative such as the one shown in (a) while others are scattered arbitrarily as shown on (b). Note the significant difference in range of the responses shown on (d), 0:14000 compared to 0:250 on (a).](image)

**Figure 14.** Filter responses of four filters applied to a dataset of 2240 image-pairs. The x-axis represents the data sample index, where the first 1120 responses are from negative samples while the last half is from positive samples. Some filters produce responses that are easily distinguishable between positive and negative such as the one shown in (a) while others are scattered arbitrarily as shown on (b). Note the significant difference in range of the responses shown on (d), 0:14000 compared to 0:250 on (a).

**Unique Thresholds**

However, since most samples produce duplicate response values, the actual feature pool is much less. For example, consider the filter response shown on graph (a) of Figure 14. The range of responses in this case is between zero and 250. Knowing that
the filter responses are always integer values, the maximum number of unique thresholds to consider is 250, which is considerably less than the number of total responses. Even then, the total number of features considered in each round of these experiments was greater than 33 million. For this reason, the training stage is the single most time consuming part of the system, taking nearly 3 hours to complete a round that produces a single feature.

**Static Thresholds**
One solution to the above problem is to fix an optimal threshold for each feature, prior to training. Therefore consequent training rounds are only required to evaluate the performance of the filter-set, rather than filter-threshold combinations. Initial experiments were carried out using this method and a comparison is discussed in the results section (4.1.2).

**Threshold Polarity**
The features are called weak classifiers as each feature is only expected to be correct 50% of the time. In order to ensure that this assumption always holds, the threshold polarity is changed if the threshold’s error is more than 0.5. The polarity dictates whether the required class should be higher than or lower than the threshold value. Consequently, if a certain threshold states that the response of a positive sample should be less than its proposed value and receives an error of 0.75 for example, then the polarity can be changed so that a positive sample is higher than the same threshold value. Its error would then be 0.25, since the weights from which the error is calculated is normalised to one.

**Repeating filters**
Since each round of training looks at the filter-threshold combination rather than a single filter, it is possible that the same filter may be picked up at a later stage of training. However, this is not likely to have the same threshold value as before, as boosting would have had an effect on the selection of the new threshold. Therefore, these repeating filters are new unique features and cannot be seen as actual repetitions.

![Figure 15](image.png)  
**Figure 15.** An example of a repeating filter in the 14th and 28th rounds of the learnt classifier with two different threshold values, 8628.5 and 7857.5.
3.8 AdaBoost

The single feature selected at each round of training is insufficient to classify persons correctly on its own. Therefore, the important task of AdaBoost is to update the weight distribution based on the selected feature. This forces the training algorithm to focus on the harder samples that were misclassified in the previous round.

In a person detection problem, the feature picked up initially is the most common characteristic of a person across the dataset, such as leg movement for example. However due to different poses and motion patterns, some of the data samples that do not have this information will inevitably be classified as negative. AdaBoost then updates the weights such that the persons missed in that round receive a higher weight, so that the features that do not detect them will receive a higher error. The feature selected in this round therefore will be forced to look for clues elsewhere in the image disregarding the legs. As a consequence, similar features (in type, size or position) are unlikely to be picked up in consecutive rounds. This helps build a comprehensive classifier that looks for different features.

Each feature receives two ratings Alpha and Beta based on its error for positive and negative classifications. The weights of the misclassified samples in this round of boosting, are increased exponentially using the Alpha and Beta values. This forces the features selected in the following round to focus on the 'harder' examples. The weights are normalised such that it is a distribution.

\[
\text{Alpha} = 0.5 \ln \left( \frac{\text{true positives}}{\text{false positives}} \right)
\]

\[
\text{Beta} = 0.5 \ln \left( \frac{\text{true negatives}}{\text{false negatives}} \right)
\]

\[
D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha y_t h_t(x_i))}{Z_t}
\]
3.9 Validation

Validation was performed using eight frame pairs for which ground truth information was added manually. The frames were sourced from several PETS\textsuperscript{1} datasets, clips from i-LIDS\textsuperscript{2} image library as well as some new video footage.

![Example validation frame pairs](image)

The classifier learnt up to the point of each validation will be used in a function similar to the final detector to scan the validation frames and return the response values. These values are examined to find the lowest value of the positive regions specified by the ground truth. The accuracy is then checked using this value to see if the desired level of detection rate is achieved while eliminating an acceptable amount of false positives.

Numerous experiments were performed, where the minimum detection rate and the minimum reduction in false positives were adjusted to find optimal values. The final values of 95% detection and a false positive rate reduction of 5% were chosen since it offered the best balance between detection time and accuracy.

\textsuperscript{1} PETS: Performance Evaluation of Tracking and Surveillance

\textsuperscript{2} i-LIDS: Image Library for Intelligent Detection Systems
3.10 Cascaded Architecture

Given the complexity of the problem, a single classifier with a large number of features would take far too long to evaluate each candidate. Instead, the classifiers will be strategically split into a cascade, placing the classifiers with fewer features at the beginning. If the candidate does not satisfy the initial stage, further processing will be stopped. This will accelerate the detection stage by eliminating unnecessary processing.

![Cascade architecture of the detector. Image source: [1]](image17)

3.11 Scale Invariance

To achieve detection at multiple scales, the training samples will be scaled to a base resolution of the training data, 12 by 36 pixels.

At the detection stage, these classifiers will operate on each level of an image pyramid calculated initially for each image set, until it reaches base resolution. Alternatively, these classifiers could operate on a fixed size detection window that scans through scaled versions of the full frame. Both methods are investigated and evaluated.

![A larger frame produces detection windows of smaller size (shown on top) while scaled, smaller frames produce larger detection windows (shown on bottom left)](image18)
3.12 K-Means Clustering

3.13 Bounding Box Resetting
4 Experiments and Results

4.1 Training Results

4.1.1 Effect of Boosting

Each round of training only requires 50% accuracy from the chosen filters. Boosting then ensures that the next training round concentrates on the misclassifications of the previous. Figure 19 shows the response of the first two feature selected by the training algorithm. A notable difference of typical responses is apparent on graph (a) reflecting on the different parts of the dataset:

Region 1 (1 to 1084) : Moving negative data
Region 2 (1085 to 2640) : Static negative data
Region 3 (2641 to 4380) : Moving pedestrians (DC random pose pedestrians)
Region 4 (4381 to 5280) : Static Pedestrians (CBCL front/rear pose pedestrians)

The first filter, which looks at roughly the head area, sets a suitable threshold that correctly classifies most samples. The misclassifications are predominantly in regions one and three. On the next selected feature, the response distribution shows that it makes a clear distinction between these regions, disregarding regions two and four which were correctly classified at the previous round.

This effect is made possible by increasing the error weights of the misclassifications and decreasing the rest, so that the selected feature would not lose much by classifying them incorrectly.
4.1.2 Static vs. Dynamic Threshold Training

Experiments were carried out to investigate the performance of static threshold training in order to reduce training time. Each filter was assigned a fixed threshold before training begins, which was not updated thereafter. The initial threshold was obtained by performing a single training round as described above, using the initial uniform weight distribution.

There was a significant reduction in the feature pool from 33,303,913 to 33,246. This was also reflected on the training time which dropped to approximately 65 seconds per round on average from more than 3 hours per round. The results were also acceptable, with the classifier achieving relevant detection and false positives targets. However, the number of features required to reach the targets was much higher than that of the dynamic threshold training.

![Figure 21. Performance of static threshold training (blue) and dynamic training (red). The graph shows the false positive rate (y-axis) against the number of training rounds (x-axis). Dynamic threshold training achieves a false positive rate of 0.03 in 39 rounds while static threshold training barely reaches 0.1 in 500 rounds.](image)

4.1.3 AdaBoost Versions

AdaBoost was implemented using the three update rules, discrete, real and gentle. From the experiments on a small dataset using only the static features, it was evident that real AdaBoost out-performed the others both in false positives and detection rate measures. Figure 22 shows the detection and false positive rates gradually reaching the expected levels with every round.

The results on Figure 22 also show the significant improvement on performance when motion filters are added, even with a training dataset of half the size. This is a positive indication towards proving the hypothesis of the project. However, since the detection and false positive rates were achieved in just 10 rounds, the individual performance of each version remains inconclusive on (c) and (d). The results on (a) and (b) show a clear indication of the difference among versions since training was performed for a longer period as compared to the motion based classifier. The first few rounds of the static filter also show the same fluctuations apparent in the motion results.
4.1.4 Static Classifier

The first few features selected for the static classifier are shown below.
4.1.5 Fusion-based Classifier

The first stage of the cascaded classifier consists of six features shown on Figure 24. Four of the six filters are motion filters suggesting the higher importance compared to appearance based static filters. The very first feature is a C type rectangle filter acting on the ∆ image with a size, shape and position roughly matching the head of a person. This is reasonable as the head is present in all positive samples as opposed to other regions like legs which can have varying results based on the camera angle.

![Figure 24. The six features in the first stage of the cascade, along with their filter responses and threshold value.](image)

The false positive rate of the classifier reduces significantly in the first few stages and then stabilises to a very low value.

![Figure 25. False positive rate of the fusion based classifier tested on the validation frameset. The rate reaches 0.1 in less than 50 rounds.](image)
4.2 Detection Results
4.2.1 Cascading

(a) After 8 stages  
(b) After 10 stages

(a) After 15 stages  
(b) After 23 stages

Figure 26. Each stage of the cascade removes a significant amount of false positives.

4.2.2 Scale invariance

Scale invariance at the detection stage was achieved by scaling the full frame rather than the detection windows. Experiments with both methods showed that the former was much faster since the integral image and motion images could be calculated once for each scale rather than individual detection window. It was made sure that there was no reduction in the number of windows considered in doing so.

The full frames were resized to 0.8 from their original size until the height reaches base resolution of 24 pixels. The number of detection windows considered at all scales and positions for a 720 by 576 size frame is 262774.
4.2.3 Results of Clustering

Figure 27. Clustering multiple detections. The first image shows the original detections that appear cluttered with too many detections around the same person. The second image is the cleaned up version using K-Means clustering. Note that some of the persons correctly detected initially are now missed as its detection was assigned to another person nearby. This offers a trade-off between a clear image and missed detections.
4.2.4 Results of Bounding Box Resetting

Figure 28. Resetting the area within the detection window to the original frame improves the display quality of the detection. However, this does not affect the actual number of detections made. This process requires that the detections are sorted according to size so that smaller detections will be cleared by the larger ones as a negative consequence.
5 Conclusion

5.1 Failure Modes

- The system implemented is only valid for full body persons and not designed to work with partially occluded persons. This also includes heavily crowded scenes in which case multiple humans are detected as a larger detection or several multiple detections around the relevant region. For example two persons close together may be taken as legs of a larger person if the rest of the image contains other motion/appearance information that simulates the rest of the body.

- The system only works with upright persons. It will not detect humans rotated with and angle of more than 10 degrees from the vertical axis.

- The system has low accuracy in adverse weather conditions such as rain and snow.

5.2 Improvements and Future Work

- Using additional feature types such as edgelets.

- Compiling a customised dataset to better suit this application.
6 References


7 Appendix

7.1 Project Evaluation

7.1.1 Mapping Progress to Project Specification Form

The following is the original text from the project specification form with the progress added to each task with a reference to the relevant sections describing them:

PROJECT AIMS:

With the increased amount of surveillance systems being implemented around the world, the need for automated detection and tracking software has rapidly grown. Among these, person detection is an important task in many applications ranging from safety and security to leisure and sports. But this task is difficult due to variable appearance of the same object category as well as varying relation with the background. Various types of clothing, weather conditions as well as different styles of walking and other human gestures and poses make this a complex problem. Related work in the field suggests it is possible to detect these characteristics reliably when considered individually but requires lengthy analysis to achieve accuracy.

An alternative way of solving the problem is to fuse together information from multiple detectors such as appearance and motion. The various sources will work together to validate each other’s results and eliminate false positives dramatically, without exhaustive analysis of individual features. The project aims to develop an algorithm that will use machine-learning techniques such as AdaBoost to build a classifier with minimal input from the user. Effectively the solution will be trained by a given dataset and then adapt itself to suit any scenario, regardless of scale, clothing and weather conditions such as rain and snow.

METHODOLOGY:

The method is broadly based on work carried out by Viola and Jones on “Pedestrian detection using patterns of motion and appearance” (2003). Accordingly, the aim is to develop a learning algorithm that will train a classifier by examining a large set of given examples. This will be based on AdaBoost described by Schapier and Singer (1999). Once the training stage is complete, persons on a given video clip can be detected using the learnt classifiers in a scale invariant manner.

Motion Detection

Detection of motion patterns is carried out using rectangular filters operating on consecutive frame pairs of the video. The presence of motion can be determined by calculating the absolute difference between two frames. The rectangular filters will then operate on shifted versions of the two frames to find the direction and magnitude of motion. There are four main types of filters that will be used:

1. Appearance filters
2. Motion direction filters
3. Motion shear filters
4. Motion magnitude filters

**COMPLETED (Page 12, Section 3.5)**

**Training**

The filters can take any scale, aspect ratio and may be positioned anywhere within the detection window. This will produce a large set of filters that can be selected by the learning algorithm to match the criteria of the desired results. The selected filters with an appropriate threshold value will determine whether a given example is positive or negative. Each of these thresholded filters is called a feature and a thresholded sum of features makes up a classifier. Building these classifiers is the task of the learning algorithm using AdaBoost at the training stage.

**COMPLETED (Page 12, Section 3.5)**

**Cascade Architecture**

Given the complexity of the problem, if a single classifier with a large number of features was used, it would take far too long to evaluate each candidate. Instead, the classifiers will be strategically split into a cascade, placing the classifiers with fewer features at the beginning. If the candidate does not satisfy the initial stage, further processing will be stopped. This will speedup the detection stage by eliminating unnecessary processing.

**COMPLETED (Page 3.10, Section 3.5)**

**Scale Invariance**

To achieve detection at multiple scales, the training samples will be scaled to a base resolution (20 by 15 as recommended by Viola and Jones). At the detection stage, these classifiers will operate on each level of an image pyramid calculated initially for each image set, until it reaches base resolution.

**COMPLETED (Page 19, Section 3.11)**

**Fusion**

The important task of AdaBoost is the fusion of motion and appearance by selecting the right balance of motion and appearance filters that yield a high detection rate and a low false positive rate.

**COMPLETED (Page 24, Section 3.5)**

**PROJECT MILESTONES**

Deliverables:

- Collection of filters – 05/11/2007
- Motion based detector – 14/12/2007
• Fusion based learning algorithm – 14/01/2008
• Cascade of classifiers – 04/02/2008
• Scale invariance – 03/03/2008
• Complete code listings of tested system – 21/04/2008

Documents:
• Final project report – 21/04/2008
• Environmental Impact Assessment – 21/04/2008
• Risk Assessment – 21/04/2008
• Presentation Slides – 21/04/2008

REQUIRED SKILLS/TOOLS/RESOURCES:

Tools and resources:
• Matlab Software
• Dataset of walking humans for training
• Video conversion, compression/decompression software

Skills:
• AdaBoost implementation techniques in the context of pedestrian detection.
• Knowledge of video / image processing techniques such as filtering, scaling, normalising etc.
• Statistical analytical skills to evaluate the performance of the system.
7.2 Final Cascade of Classifiers

Stage 1

Number of features = 6

Classifier threshold = -3.509876

Features:

<table>
<thead>
<tr>
<th>Features</th>
<th>Threshold 1</th>
<th>Threshold 2</th>
<th>Stage</th>
</tr>
</thead>
<tbody>
<tr>
<td>5638 3 3 3 1 9 2</td>
<td>0.822034678201859490 0.780538420370329940</td>
<td>266.500</td>
<td>1</td>
</tr>
<tr>
<td>30669 3 2 17 1 9 6</td>
<td>0.593862818403763870 0.477780933015220450</td>
<td>22196.500</td>
<td>-1</td>
</tr>
<tr>
<td>12023 4 3 5 1 4 18</td>
<td>0.508535761095563640 0.720986013560712120</td>
<td>62.500</td>
<td>1</td>
</tr>
<tr>
<td>9423 4 1 3 5 6 16</td>
<td>0.754805126483844240 0.419262356514580210</td>
<td>538.500</td>
<td>1</td>
</tr>
<tr>
<td>32475 1 3 19 5 2 4</td>
<td>0.378119660277098700 0.556315397784621020</td>
<td>8.500</td>
<td>1</td>
</tr>
<tr>
<td>6678 6 7 3 1 4 14</td>
<td>0.389245774705578750 0.554992521743723160</td>
<td>271.500</td>
<td>1</td>
</tr>
</tbody>
</table>

Stage 2

Number of features = 6

Classifier threshold = -2.604710

Features:

<table>
<thead>
<tr>
<th>Features</th>
<th>Threshold 1</th>
<th>Threshold 2</th>
<th>Stage</th>
</tr>
</thead>
<tbody>
<tr>
<td>27952 3 1 13 7 3 8</td>
<td>0.361232791072828130 0.503371826260066600</td>
<td>6663.500</td>
<td>-1</td>
</tr>
<tr>
<td>4649 4 3 1 7 2 6</td>
<td>0.41466762169537430 0.535706767256833190</td>
<td>8.500</td>
<td>1</td>
</tr>
<tr>
<td>32233 1 2 19 3 2 4</td>
<td>0.407909739414041850 0.46148526004155680</td>
<td>32.500</td>
<td>1</td>
</tr>
<tr>
<td>8765 4 2 3 5 4 4</td>
<td>0.39568638659545440 0.394360471994310680</td>
<td>51.500</td>
<td>1</td>
</tr>
<tr>
<td>31605 4 3 17 7 2 6</td>
<td>0.426260296691509610 0.379586898717412520</td>
<td>13.500</td>
<td>1</td>
</tr>
<tr>
<td>5270 7 5 1 9 2 4</td>
<td>0.426079032504473990 0.330198373597808010</td>
<td>85.500</td>
<td>-1</td>
</tr>
</tbody>
</table>

Stage 3

Number of features = 5

Classifier threshold = -2.100743

Features:

<table>
<thead>
<tr>
<th>Features</th>
<th>Threshold 1</th>
<th>Threshold 2</th>
<th>Stage</th>
</tr>
</thead>
<tbody>
<tr>
<td>587 4 1 1 1 4 8</td>
<td>0.315887013983631840 0.440760055526870810</td>
<td>76.500</td>
<td>1</td>
</tr>
<tr>
<td>33058 3 2 21 5 3 2</td>
<td>0.514439870338610050 0.348999074528955500</td>
<td>8628.500</td>
<td>-1</td>
</tr>
<tr>
<td>10426 4 3 3 9 2 10</td>
<td>0.380564639716528320 0.467324477164994030</td>
<td>13.500</td>
<td>1</td>
</tr>
<tr>
<td>18808 4 2 7 7 2 14</td>
<td>0.404282074617464460 0.423045808279702450</td>
<td>71.500</td>
<td>1</td>
</tr>
<tr>
<td>33002 4 3 21 3 6 2</td>
<td>0.259053599515888020 0.420623437260758460</td>
<td>6.500</td>
<td>1</td>
</tr>
</tbody>
</table>
7.3 Validation Frame Pairs

(a) PETS2

(b) PETS2

(c) PETS2

(d) Crowded Scene

(e) Football Match

(f) Victoria Station

(g) Westminster Station

(h) Westminster Station